Team Juggernaut's Robust Approach for the 2007 DARPA Urban Challenge

Submitted June 1, 2007

George A. (Troy) Takach, Jr.
Team Leader, Senior Managing Partner
DesignJug
2999 Ksel Drive
Sandy, Utah 84092
troy_takach@designjug.com
Mobile: 801-870-8470

Thomas Grover
Communications & Marketing Manager
Kairos Autonomi
508 West 8360 South
Sandy, Utah 84070
thomas_grover@kairosautonomi.com
Mobile: 801-318-1289

Having developed a proven hardware platform and a robust software base from Grand Challenge 2005 technology, DesignJug approaches the Urban Challenge 2007 with an almost exclusive focus on sensor and software development as it solves the complex challenges of autonomous urban driving.

Beginning with the Pronto4™ Strap-on Autonomy System — a retrofit kit that makes any existing vehicle with a steering wheel a drive-by-wire system — and including a flexible, shared variable software base running on an open architecture system, DesignJug is enhancing its technology by creating a new sensor platform with supporting software that provides input to the team's Actively Refined Reality Model.

This paper also outlines the following elements that support DesignJug's efforts:

- using a spiral model development paradigm for rapid creation and assessment
- implementing complex navigation and traffic management
- executing an aggressive and frequent test schedule to achieve critical tasks
- combining several compatible tasks to push progress forward
- utilizing software workbenches that contain all required software and vehicle infrastructure so software engineers can focus on the application or problem at hand, not the architecture or language of the system

These key elements form the core of DesignJug's efforts to successfully compete in the Urban Challenge.

1. INTRODUCTION AND OVERVIEW

1.1 Introduction

Team Juggernaut competed in DARPA's Grand Challenge 2005. The team was able to start with absolutely nothing and deliver, in a 10 month time frame, an autonomous vehicle that ultimately traveled 63 miles over the Grand Challenge course. Since that time, the technology has been migrated to a variety of commercial vehicles, including its 2007 Urban Challenge entry – a Jeep Liberty. All key players from the 2005 team are participating on DesignJug's Team Juggernaut for the Urban Challenge, and many of these team members have been working full-time since then to advance DesignJug's autonomous vehicle technology for the Urban Challenge.

1.2 Overview

In order to compete in the Urban Challenge, vehicles must maneuver "in a mock city environment, executing simulated military supply missions while merging into moving traffic, navigating traffic circles, negotiating busy intersections, and avoiding obstacles."[1] DesignJug has already had a great amount of success in demonstrating autonomous ground vehicles driving in an urban area in the presence of vehicular traffic.

The team's basic approach to solving this complex situation is to create a virtual birds-eye view where all required pathing for the autonomous vehicle is seen from a high vantage point. The vehicles and all observed obstacles can be placed on a storyboard statically or with their changing trajectories. If time is stopped, all possible paths to meet the objective can be drawn out, the best selected and then executed. The entire process is repeated continuously from this birds-eye view. Most third-person video games follow this rough approach while imposing real-time pressure.

Once the birds-eye view has been created, the view is populated with observed objects, the software progressively identifies those objects and their trajectories, and the field is analyzed. Obstacles can be observed in real-time using unique cameras systems. Optimum pathing is developed using a number of various shaped-based techniques. Multiple paths are generated and weighted based upon parameters such as length, clearance, number of turns and the severity of the turns, currently routed path, etc. The best path is chosen and executed. This process is repeated 10-15 times per second.

The core of the development program was centered on a test schedule. Successful completion of these tests is the goal of the development effort and the assigned tasks. Tasks have been developed from a known need point-of-view, with a few of an exploratory nature, all supporting the test schedule. The test detail and schedule are matched to the program schedule.

1.3 Spiral vs. Waterfall Development Paradigm

Two major development paradigms exist for the design and delivery of complex systems. The waterfall model defines a method of project stages that directly lead from one stage to another where work is performed in rough sequence — design leading to validation, leading to build, leading to testing, leading to re-design, and so on. This can be a lengthy process but is easy to document as to what comes next. Large teams can follow this model because of the documented path. Exact delivery time is very difficult to predict using this approach because of its linearity.

The spiral model follows a path that always has a deliverable at increasing degrees of performance against a functional goal. Prototypes of functions and systems are rapidly created, spun into existence, assessed and iterated in small increments toward 100% compliance with specifications. The functionality of that prototype is assessed against the end goals of the product. A list of tasks is identified that are required to reach the next goal. Tasks that identify new elements of the product are spiraled as well, tested and reviewed. Man learns by doing. If the end goal is known but the path is uncertain, the team learns along the way. Using the spiral model, a program manager is never without a deliverable and the deliverable progresses towards full compliance with desired specifications. This approach works well with smaller focused teams, and our approach for the Grand Challenge and Urban Challenge has followed this model.

Since the majority of the work to be performed for the Urban Challenge is software, particular attention was given to the system design to make it easy for DesignJug to leverage team members that have very vertical skills. Specifically, the shared variable (SV)/Pipe architecture makes it easy for a software design element to be broken into specific sized chunks and then the interface to that "chunk" is easily identified (SV and pipe contents) and simulated. The software team member can then design, create, test and deliver that "chunk" back to software assemblers for integration into the entire system.

1.4 Technology Framework

Preparing for the Urban Challenge is a large, multi-faceted program with many projects. Most projects can be broken down into small tasks that can then be planned, accomplished and tested. Although at the top level an individual can define a specific goal and design approach to achieve that goal, he must be flexible enough to adjust the course of the project or even the program if success is not occurring in a timely manner as expected. The structure of the entire system must be designed such that design change can occur at almost any level without impacting the performance or stability of another level or component. This naturally leads to a compartmentalized approach where discretive modules can be identified, developed and tested in isolation and then brought into the whole for integration and further testing.

The entire computing platform and I/O pathing are built on an open architecture (OA) system. Using Windows XP® as the operating system and USB 2.0 as the primary I/O channel, all software is written in common, low-skill based languages — this provides the ability to focus on the application, not the language. By limiting the usage of special hardware or sole source hardware for the OA platform, we can take advantage of the continual advancement of the industry. All application software runs as individual executables sharing data among the executables and the hardware system. Visual Studio® 6 languages, a stable language set, are chosen as the primary software languages for development. Software components such as ActiveXs and DLLs (Dynamic Link Library) can be purchased on the market at advanced version levels, in most cases version 4 and above.

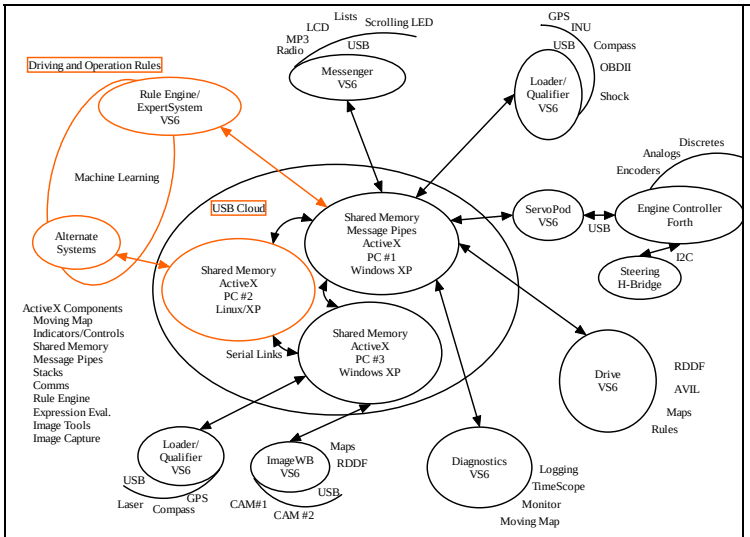


Figure 1, this diagram shows how all inter and intra-communications between applications and hardware on the computing platform can be monitored and exploited as required in real-time. Because of this standardization, a third-party tool is used to automatically instrument the code and generate performance data on every line of code. Significant performance enhancements are possible and realized by simply addressing the identified portion of code. Another third-party tool is incorporated for coding style validation. It checks all written code for specific classes of coding style problems, flags those areas and generates a report. These can be used to assist in the adjustment of the specific programmer's style and clean up delivered code.

A shared memory structure with application data pipes was selected as the primary software structure. Shared variables (SV) are created in an 8MB area within Windows XP. All applications can read and write values to and from specific type, defined variables. These SVs create a real time data dictionary that is managed by the application programs. The SVs contain dynamic, but snapshot, values defining the vehicle system and its environment. Named data pipes assigned to each running application are used to communicate sequential information between those applications.

2. ANALYSIS AND DESIGN

The following paragraphs discuss the programs, components and previously completed work that support DesignJug's Urban Challenge effort. The discussion includes accomplishments from the Grand Challenge as well as programs that were carried forward after the completion of the Grand Challenge and current efforts. DesignJug gathered, developed and created many supplies and support materials that directly support the creation and testing of autonomous vehicle technology.

2.1 Technology Base

The work DesignJug has done is founded on a several key principles that form the basis of the team's approach. DesignJug has developed reproducible, universal retrofit mechtronics for the drive-by-wire portion of the autonomous ground vehicle to leverage the infrastructure of existing vehicle platforms. The Urban Juggernaut, DesignJug's vehicle, is designed to rely primarily on passive sensors such that many of these systems can operate together without interference. An explanation on the sensors and systems follows.

For the Grand Challenge, Team Juggernaut developed and fielded technology that performed Global Positioning System (GPS) path following at speeds greater than 30 mph. The system includes vision-based obstacle avoidance and vision/object based pathing using a moving map and objects placed on the map. Team Juggernaut also developed an approach for portable drive-by-wire implementation. The Grand Challenge vehicle, Desert Juggernaut, did not detect hay bales at the 2005 NQE, nor had Team Juggernaut worked with a tunnel prior to that time. The combination of the tunnel and hay bales proved to be the team's nemesis.

Returning to the Grand Challenge course in late November 2005, the Desert Juggernaut ran 63 miles with largely the same system that was used on race day. A failed alternator coupler brought the vehicle to a halt. Team Juggernaut created and used a DARPA compliant E-Stop system for that run.

The Pronto4 Strap-on Autonomy System is a proven drive-by-wire hardware platform that DesignJug is using in the Urban Challenge. It was spun out of the Grand Challenge as a retrofit kit for existing vehicles, and has been tested on a number of vehicles, including sedans, minivans, trucks and SUVs. The Pronto4 system has been released to the market and has received favorable reviews, with a number of systems being sold. It will also be used by two other Urban Challenge teams. The Pronto4system can also perform in dual-use environments driven by a man in normal capacity or machine.

Three primary vehicle actuators are used on the Pronto4 system. The steering actuator turns the steering wheel through 900 degrees of travel. Steering actuation is limited to +-425° to save wear on the vehicle power steering system. System latency from the Operator Control Unit (OCU) to onset of actuator performance is typically less than 150ms. Full lock-to-lock steering can occur in less than 2 seconds. The steering actuator can deliver up to 50 ft-lbs. without power steering when required. Nominally 3-5 ft-lbs. are required. Full brake and throttle actuation, 0 to 100%, is less than 500 ms. Transmission shifting occurs from Park to Drive in less than 1 second, although the transmission may respond slower based upon vehicle mechanics.

Figure 2, the **Pronto4™** steering wheel system, installed in an indigenous vehicle. Note the single point compliant attachment to the vehicle and normal room for a vehicle operator.

The Pronto4 system runs from a single 12 or 24 vdc power source. Power draw is nominally 5 amps while not driving or running actuators, under 20 amps running actuators without the engine running and under 10 amps running actuators and the engine. The power module contains a secondary battery with a high-speed <25 us isolation switch. A simple connection is made to the main vehicle battery with 6 awg cabling from the power module. All battery charging occurs

from the main battery. The Pronto4 is quickly isolated (and thus protected) from the main vehicle power when main battery voltage drops below nominal levels, such as vehicle start, winch operation or other high draw main vehicle applications occur. 12 vdc is most common on vehicles, but other voltages are available.

Operations and user interfaces for the Pronto4 system are controlled by software running on a PentiumM 1.6Ghz or better CPU under Windows XP. User interfaces are all graphical and intuitive in nature. All vehicle operations can be tested by operation of these graphical user interfaces (GUIs). Sensors and actuators use either standard serial (RS-232/422 etc.) or USB 2.0. Most common interfaces are available such as Ethernet, parallel, FireWire, VGA, keyboard, mouse, etc. Multimedia hardware and drivers are expressly disabled and not loaded on the Pronto4 system hardware. Where possible, Internet access has been restricted or eliminated. These simple modifications/restrictions bring XP into the domain of deterministic real-time operating systems. XP at this point is sophisticated thread manager with a significant inter-process communication infrastructure.

Control of the Pronto4 system can occur from a number of sources. All Pronto4 system operations are controlled via shared variables and data pipes. These shared variables reside at the Windows OS level and can be accessed by any executable. Using the SharedLink protocol, access to Pronto4 system operations occurs over UDP or serial links accessing any and all shared variables. Less than 25% of the primary Windows computer is used for operations; therefore, user programs can reside on that machine with access to the shared variables. For example, to change steering angle, simply write the steering angle to the VEH_STEER shared variable, or change a value from 0-100% for throttle and brake to VEH_THROTTLE and VEH_BRAKE. JAUS RA 3.2/3.3 can also be used to access these shared variables.

Simplicity of installation is addressed on many fronts with the Pronto4 system. Less than 20 cables are required for installation in vehicles using the Pronto4 system in tele-operation or semi-autonomous modes of operation. The majority of the cables are 4 conductor shielded used for communications, 19 conductor cables for signaling and 2 and 4 conductor cables used for power.

Other specifications include:

- Capable of standard driving speeds up to 60 mph
    - City, highway, off-road
- Weighs less than 100 lbs.
    - No special equipment is needed for transporting the kit
    - Minimizes shipping costs
- Open architecture, Windows-based interface
- 900° lock-to-lock steering wheel angle (+/- 450°)
- Steering response time (lock-to-lock) when vehicle is in motion < 2 sec.
- Brake and throttle response time of command (0-100-0%) < 1 sec.
- Command latency approximately 150 ms
- System operates via a 115.2K baud stream
- Ancillary functions include lights, turn signals, and horn
    - Provides for RPV, RGV use after dark
    - Conforms to visual and auditory signals expected by drivers in manned vehicles

- Optional ancillary functions available
    - Door locks, wipers, windshield fluid spray and electric windows
- Fits in a 24” x 20” x12” envelope, uninstalled
    - Minimizes space required inside the vehicle/vessel

2.2 Primary Sensor Approach

To achieve the criteria for the Urban Challenge, DesignJug has needed to significantly improve the Urban Juggernaut’s sensor platform, as well as the ability to correlate and train that platform. Based primarily on vision systems using Laser Imaging Detection and Ranging (LIDAR) for training, DesignJug has built modular vehicle instrumentation platforms using standard vehicle racking sytems (Thule or Yakima). These instrumentation racks are mounted on the top of the Urban Juggernaut and other test vehicles and are not vehicle specific. This many test platforms are available not just our race vehicle. Sensors include:

- Two to Four (4) proximal and one (1) central multi-mode, omni-directional cameras
- Two (2) to four (4) laser scanners, 180 degrees, and one laser range finder
- Two (2) color cameras

This platform will be used for all localization, navigation and environment observation of and around the vehicle. There is a front, rear and side mounted ultrasound ranging system for close field identification of less than 10 feet. There are front and rear mounted strip cameras that will observe the ground immediately in front of and behind the vehicle for ground mark verification. In addition, an existing dual antenna standard GPS (high availability), differential GPS (low availablility), magnetic compass, gyro-stabilized INU, and other sensors are located in or on the vehicle.

Figure 3, shows the **sensor suite** on the Urban Juggernaut. Additional sensors are located on the sides, front and back of the autonomous vehicle.

2.3 Camera System

A 180 degree laser fan on tilt platforms, called a laser autoranger, is created using a SICK laser scanner. Initial work is done with a single forward facing laser autoranger an additional rear autoranger may be added prior to the NQE.

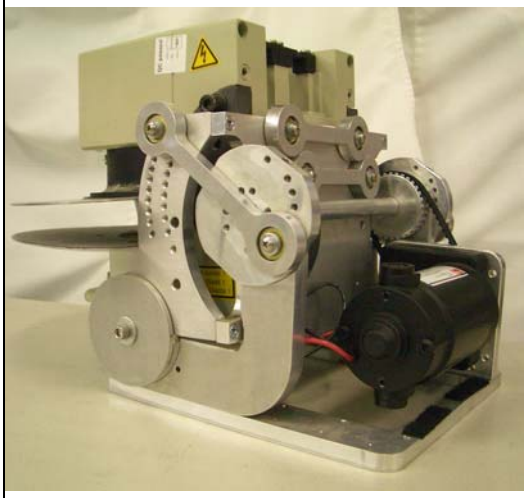


Figure 4, is the **Laser Autoranger** that DesignJug designed. It can tilt dynamically between a fixed set to start and stop angle based upon the configuration of the linkage bar. An eight-station Geneva mechanism (not shown) allows for five fixed angles to be continuously scanned or continuous angular change to occur (the figure shows the Autoranger set for continuous scan). Current actual usage is a tilt angle on demand based upon driving conditions and situation. The laser can scan up to 300 feet in a max 180 degree fan. It is limited to a 95 degree fan for practical usage. 1 to 2 of these lasers and their Autorangers are be used on the top of the vehicle and mounted in the middle of the sensor platform. They will cover 360 degrees around the vehicle from 0 to 25 degree tilt or 300 feet to 10 feet based upon tilt to earth. Twenty-five degrees looks within 3 feet of the vehicle.

The 180 degree laser fan with variable termination (approach angle) is a brute force method to measure distance to the vehicle and movement of hard objects in relation to the vehicle. Processing and proper characterization of this data is challenging but relatively straightforward to achieve. This sensor is an emitter and ultimately very undesirable to be used for a final system, but it is an excellent real-time quantification of discerned objects in the omni-view birds-eye view. The laser distance system is being used to develop and test the birds-eye view, and the laser systems will be used to train the omni-view system.

The two to four omni-view cameras create a more detailed birds-eye view of the vehicle and its surroundings. The omni cameras are placed on each corner of sensor platform located on top of the vehicle. These omni cameras are hyperbolic mirror based upon test results as opposed to the original spinning mirrors prototypes. Additionally, one omni-view camera raised in the center of the instrument grid will also provide a single, gross, birds-eye view. The fast pan 3D stereo camera is placed inside the vehicle and observes from a human point of view.

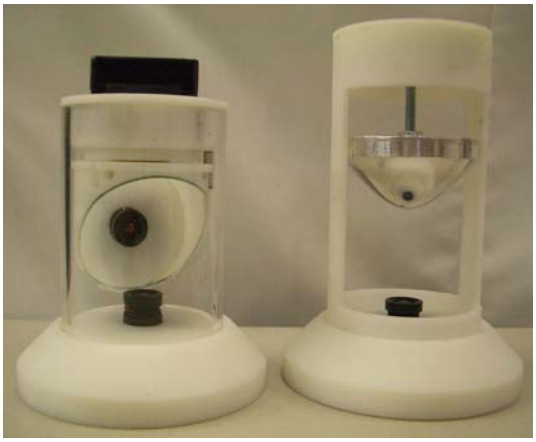


Figure 5, two types of **Omni Cameras** were designed and tested. One uses a continuously rotating mirror and collects its 360° view over a full revolution of the mirror. The image is un-rotated by software or a video processor in real-time. It has the advantage of being able to collect quite a bit more information with a lower resolution camera. There is no data loss in the collected images, but it has moving parts and a lower frame rate. The hyperbolic camera generates a 360° view of the surrounding environment in a single frame. It must be unwrapped by software for human viewing but can be used for object movement detection and quantification in the images as is. The drawback is that there is a loss of image quality close to the camera and a reasonably intensive routine is required to unwind the image. It has no moving parts and is very simple to install. These are designed to dimensional specifics (minor a and b) and aluminum evaporated at our lab. Each were evaluated and used in our vehicle testing.

Omni Camera on Instrument Rack

← Production Camera
Production Prototype Camera→

The ultimate goal is the use of a single omni-view camera and 3D fast pan camera mounted inside the vehicle. The software "view" system will need to be developed and cross-referenced in layers in order to achieve this goal. The single camera can be used to find unexpected or gross motion. The fast pan cameras can be used to refine the image areas the omni camera identifies.

The omni-view cameras are based on application-specific, hyperbolic mirrors. DesignJug's research lab has an in-house aluminum evaporator and four-axis mill for the creation and "silvering" of these hyperbolic mirrors. Mirror geometry is based on optimization of parameters between desired resolution and vertical viewing cone. Hyperbolic omni-view cameras require no

moving parts. Accurate distance measurement up to 300 feet are directly achieved using pairs of

Figure 6, the **Fast Pan Camera** is made of a pair of cameras mounted on a rapid-moving, low mass platform. This camera can pan as fast as the human head and can follow a head tracker as needed. In autonomous mode it is used to look at areas of interest (or movement) identified by the omni cameras and collect higher resolution images. The cameras operate in field sequential mode when used for tele-operations, providing the remote pilot with 3D vision. The primary usage of the 3D vision is not to give the pilot depth perception but to significantly reduce blind spots in the vehicle, which allows the fast pan camera to be placed inside the vehicle. This approach uses standard vehicle window cleaning equipment, thus increasing the operational environment of the vehicle, and it also allows for a more covert installation of the cameras. The two cameras are currently mounted on each headrest of the vehicle front seats, which does not interfere with seated driver or passenger.

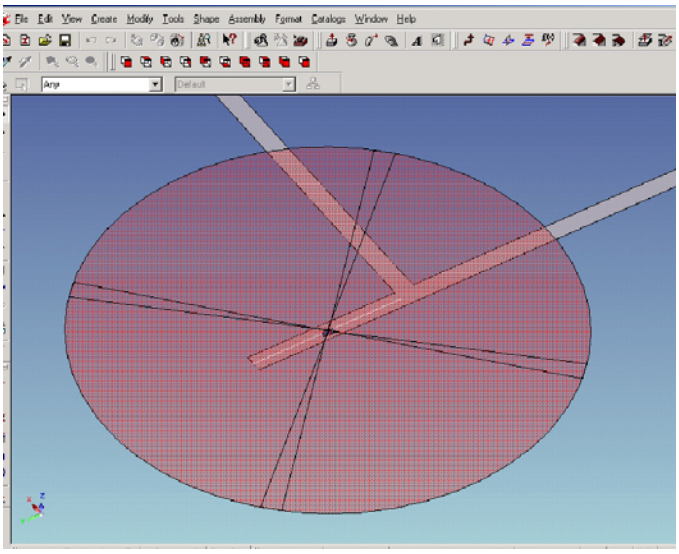
cameras. In this configuration, six pairings of cameras can be utilized.

Existing hardware quad camera multiplexing allow all omni cameras to be viewed and operated upon on the same frame. Frame capture takes place at the lowest usable resolution. Resolution scaling (320x240 to 640x480 to 1024x768) is used to reduce processing load until something interesting needs to be “focused” upon.

Once an area of interest is determined, the 3D fast pan camera observes the area in more detail and further enhance the detail in the Actively Refined Reality Model (ARRM).

Since the camera system is the primary means of object identification and navigation assistance, care is taken to assure the best possible images. Shadows are expected to cause a significant issue with obstacle or blocked road detection. A shadow camera is used help assess in real time where shadows are and their density. The cameras are dynamically adjusted based upon the data from the shadow camera.

The entire sensor system is simulated to assure proper coverage as it changes dynamically. Figure 8 shows how Team Juggernaut easily accomplished this using a programmable CAD system.



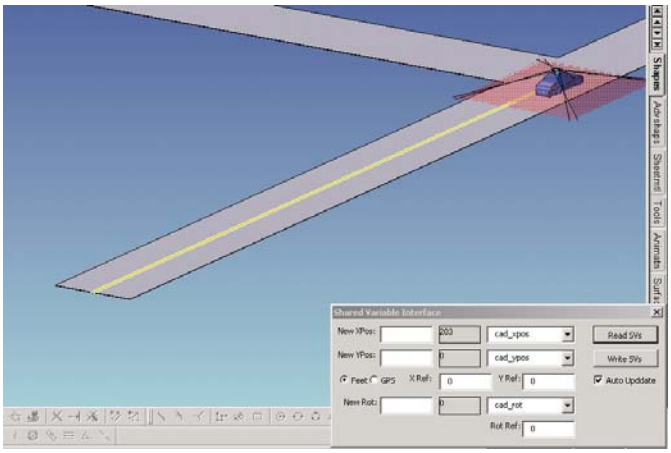


Figure 8, using IronCAD®, physical vehicle information is placed alongside actual road dimensions and physical sensor mechanics. It is then possible to see, in real-time, the sensor coverage as it dynamically changes while the vehicle drives along a known course such as that defined by a mission data file (MDF) within an RNDF. This allows the sensor fans to be optimized for the best coverage balanced against resolution. Left, the four lasers are tilted at 0 degrees with a 200 foot range. Below, the vehicle has moved along the road and the lasers are tilted to 15° intersecting the road.

2.4 Actively Refined Reality Model

This reality model is the core of the navigation and pathing of the autonomous vehicle and is actively refined using the primary sensor system. A route network definition file (RNDF) is loaded into a vehicle database along with its defined or inferred objects. It creates an observation mask that is laid down upon the mapboard. That mask is used to filter out or quantify the "noise" seen in the birds-eye view generated by the omni-view cameras. As the vehicle drives around the RNDF it observes and gains clarification of various objects as well as refines the attributes or objects defined in the RNDF. These observations are documented in the database in real-time.

2.5 Pathing

The problem of navigation from one checkpoint to the next can be solved by a reduction to a directed graph traversal. Each checkpoint is a node in the directed graph, and an edge represents a direct path between two checkpoints. In order to facilitate different route selection choices

within this framework, each edge will have significant amounts of data associated with it. The simplest model only associates the GPS waypoints that describe the path with each edge. In order to implement more complex route selection (shortest path, minimum left turns, etc.), many other attributes can be associated with an edge, including distance, number of left turns, etc.
After the RNDF has been transformed into a graph representation, it is then easy to build a table of all possible paths. The graph already represents all paths with one edge, so we build a table of all paths from two edges up to an arbitrary *N* edges. However, the search may be exhausted before the path of *N* edges is reached, so a path is considered exhausted when it crosses a checkpoint already existing within the path. Once this table is complete, it can be organized to list all paths between checkpoints *A* and *B*, sorting them by some chosen criteria.

It is possible that the computational complexity of this table generation may be overwhelming, but there are other possibilities that stem from the directed graph representation of the RNDF. For example, the actual path search could be performed using some variant of the A* algorithm with appropriate pruning filters. If necessary, a probabilistic framework could be exploited as well, where path possibilities are ranked according to their likelihood of meeting the criteria for success. These methods, or the methods described above, can be used to find an appropriate path between any two checkpoints. Also, the methods can be used iteratively to find an optimal path crossing all MDF checkpoints, in the specified order, while following traffic laws.

In this framework, route selection becomes an O(1) operation after significant preprocessing: the worst case time for generating the graph representation is $O(n^2)$, generating all possible paths as described above is $O(n!)$. However, this may not be a significant problem, as the *n* will be relatively small (likely less than one hundred) and the software does not have a significant time bound on preprocessing the RNDF since it will be done off-line. Implementing optimizations, such as chaining already-searched paths, will decrease the complexity. Similarly, the simple solution to dynamic changes in the RNDF (road blockages, etc.) will be to simply rebuild the graph and the path list with optimizations made allowing quicker and simpler recalculation.

While this approach solves the overall problem of map-based path planning, the path between two points associated with an edge on the graph does not necessarily contain enough information to successfully navigate from one checkpoint to the next; it simply contains the GPS coordinates given in the RNDF. It may be necessary to follow road markings, navigate open spaces like parking lots, or face other challenges.

Alternate approaches to pathing include the ability to treat the graphical RNDF with MDFs similar to that of a printed circuit board (PCB) that has many obstacles and routing constraints. Multiple paths are generated and weighted based upon parameters such as length, clearance, number of turns, severity of the turns, currently routed path, etc. The best path is chosen and routed and assigned a certainty number. Shape-based techniques prove to be the most general at solving complex routing problems. Team Juggernaut has expertise in PCB routing and CAD creation.

2.6 Software Base

The existing software base is very extensible. Functions can be easily enhanced and new functionality added with minimal effort and impact. All functions needed to meet the Urban

Challenge requirements are created or framed within this code base. The software base is quite visual, allowing communication among team members while easing testing and validation. These are all significant, qualified programs that perform critical autonomous ground vehicle functions. Following is a partial list of developed software programs:

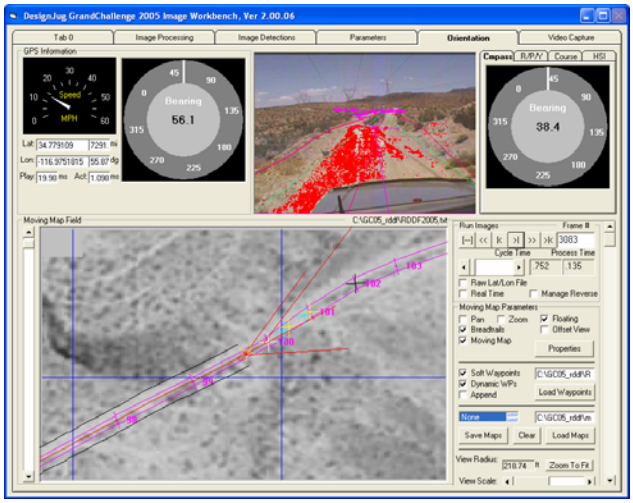
Table 1, Autonomous Ground Vehicle: Grand Challenge and Post Grand Challenge	
Grand Challenge 2005 Software	
*djLoader	Convert serial streams to SVs, qualify sensor data, perform correlation
djMessenger	Manage system messages, logging, distribute to user displays
djServoPod	Interface to vehicle FORTH based computer and actuators
djDrive	Pathing and vehicle driving, execution of scripting language AVIL
*djImageWB	Image processing and development, video frame capture, object detection
djMMViewer	Vehicle performance playback, post analysis of vehicle dynamics
djMonitor	SV monitor, strip chart, XY scope, diagnostics, logging
djMission	Mission planning – vehicles, courses, missions, RDDF editing/generation
* Modified for Post Grand Challenge Use	
Post Grand Challenge Software	
djSharedLink	Spread spectrum radio point-to-point SV link, command/control
djSharedLCD	Real-time user display of SVs and video annotation, editing of displays
djDriveByWire	Human vehicle driving interface, wheel, brake, throttle, shifting, ancillaries
djMachineI	Machine intelligence dev., SVinterfaces for neural nets and fuzzy rule sets
djSVManagement	Create, operate shared variables, SVs and named pipes, diagnostics
djSimulate	Real time birds-eye view driving, simulation testing, multiple vehicle testing
djOmniView	Management of omni-vision cameras, object detection and localization
djDrivenbyWire	Local logic operations for RCV and RGV vehicle modes, script processing
djAuxFunctions	Control of ancillary vehicle functions
djVideoHead	In-vehicle video operation management, video multiplexer, overlay text
djVideoLook	Host vehicle video operation control panel, 3D headset, head tracking
djEmulate	Real-time UGV emulation in virtual worlds, PC based targeted at Xbox 360
djPark	UGV parking management, RNDF generation/editing
djPerformance	Real-time assessment of vehicle performance and temperature monitoring

2.7 Workbenches

Extensible workbenches (WB) are created with predefined software interfaces to vehicle platforms. Each of these platform-interfaced workbenches allows the software engineer the ability to explore, develop and test application code at his/her desk prior to bringing it to the vehicle. Iterations per hour for the engineer are significantly increased. The WBs are used in the vehicle run-time environment as well as offline to develop approaches and algorithms. WBs are provided to universities and partners to focus development on the required application/vertical problem. The WBs have the ability to record and playback pertinent information.

These are two of the many workbenches that DesignJug developed for usage with our Urban Challenge effort and are now used to solve challenges associated with autonomous ground vehicle development: one is for the vision systems (Figure 9), the other is for implementation of contained machine intelligence (Figure 10). Other workbenches cover simulation, pathing and planning.

DesignJug uses the djMachineWB for “calibrationless” steering straight algorithms, control loop tuning and adaptive speed control, and the team will also explore image processing and open space path planning.



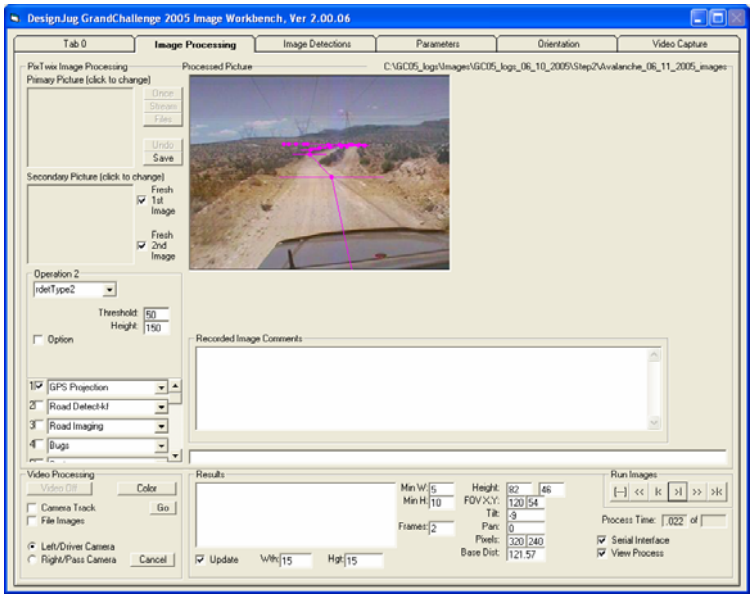


Figure 9, the **DJImageWorkbench** is used to develop image processing algorithms. It uses a series of recorded, instrumented images (video) played back through various image detection methods and operations. Shown are methods of GPS waypoint image projection, road color bucket imaging and edge of road detection, separated for clarity, all on the same section of road. The data is fused into a single display with a moving map and the course traveled. The images can be single stepped or played back at the recorded rate up to 30 frames per second. Instrumented data such as GPS, heading, speed, etc. is displayed.

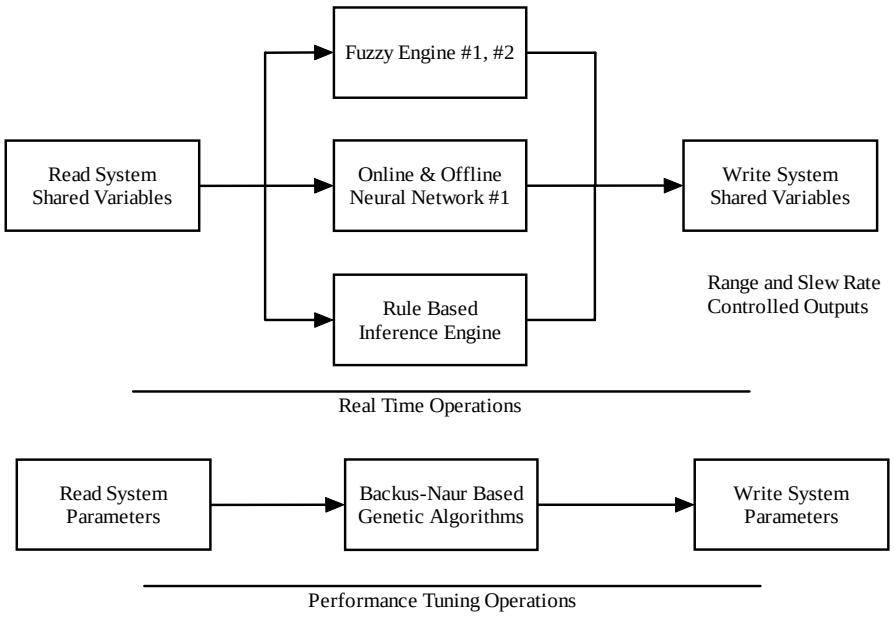


Figure 10, the **djMachineIWB** is used to develop and test nonlinear learning-based algorithms in a machine environment. Recorded information about system behavior can be played back in discrete steps or in volume. Learning or convergence can occur offline or when attached to the vehicle. All parameters and I/O can be interfaced to the vehicle through SVs data pipes. Script editing and execution occur in this WB. After a net has been trained or a rule base created they can be directly tested on the vehicle in real-time in a very contained environment.

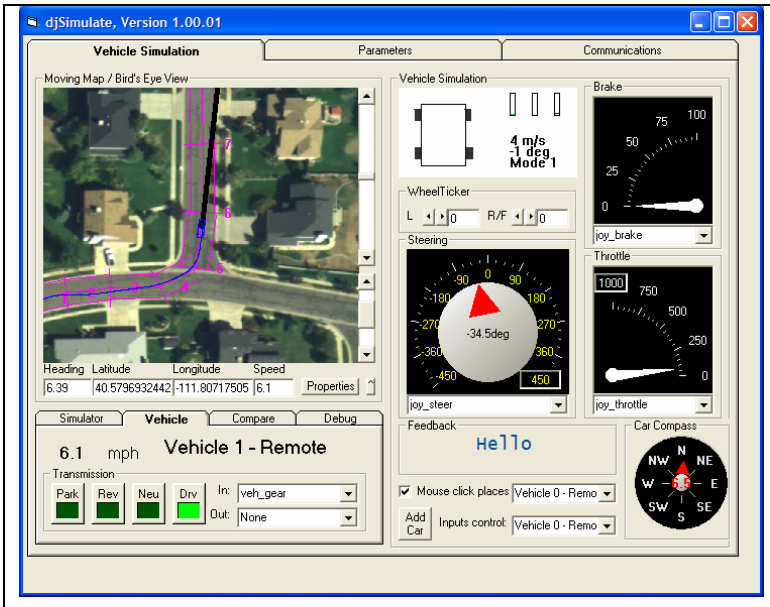
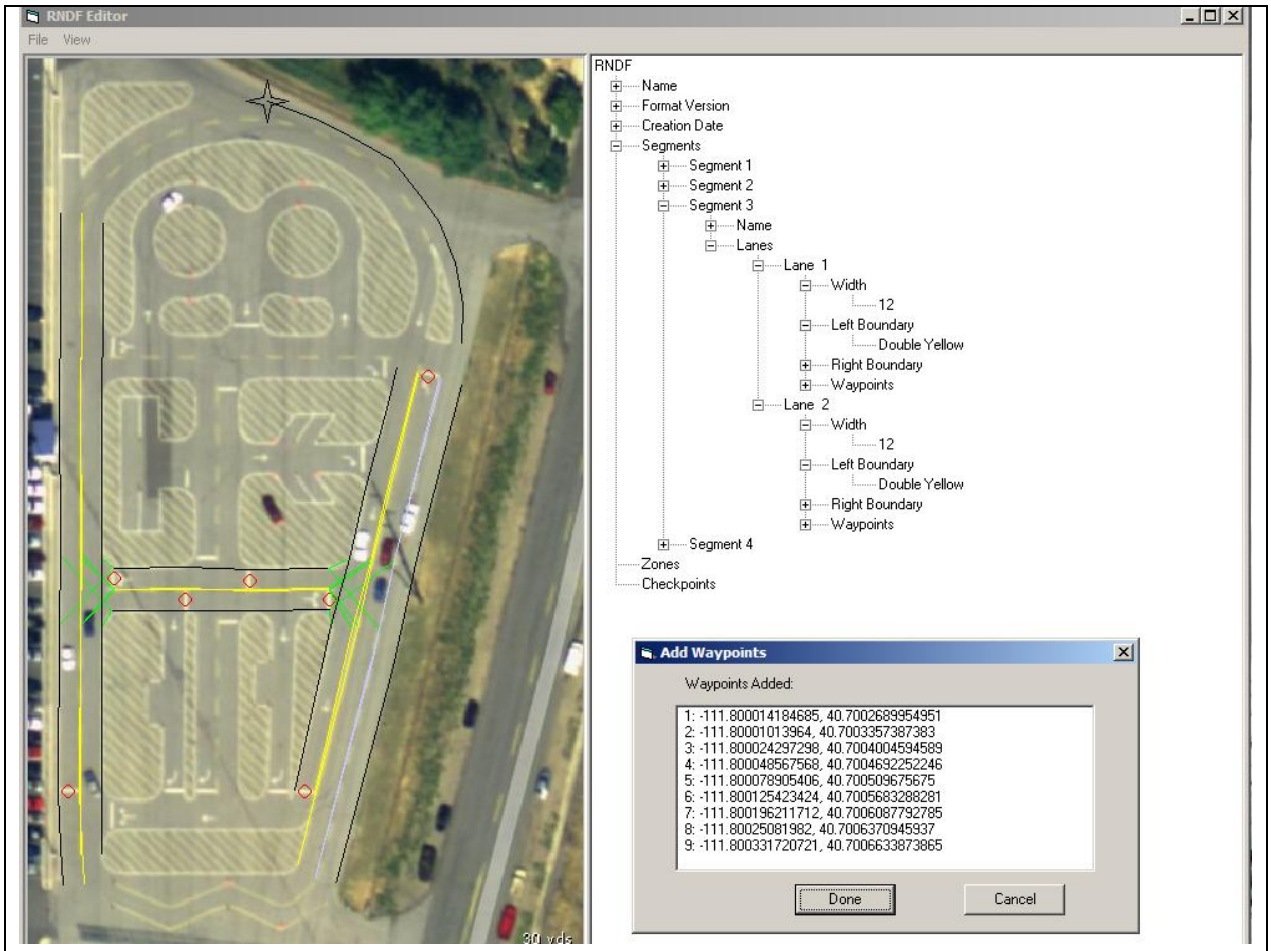


Figure 11, the **Vehicle Physics Simulation Program** is used to test the ActiveX® vehicle dynamics component that runs on the autonomous ground vehicle in real-time as part of the sensor data validation. This test environment can simulate multiple vehicles at once for following mode testing and others. The program uses or generates SVs to interface to the actual system. A birds-eye mapboard is used to place the vehicle, known route information and aerial photos, as well as other objects. This tool can be used as a source of user data or to playback recorded data from the vehicle. An external steering wheel can drive the simulated vehicle and has parameters such as weight, turning radius, static and dynamic friction coefficients, acceleration, braking distance, etc.



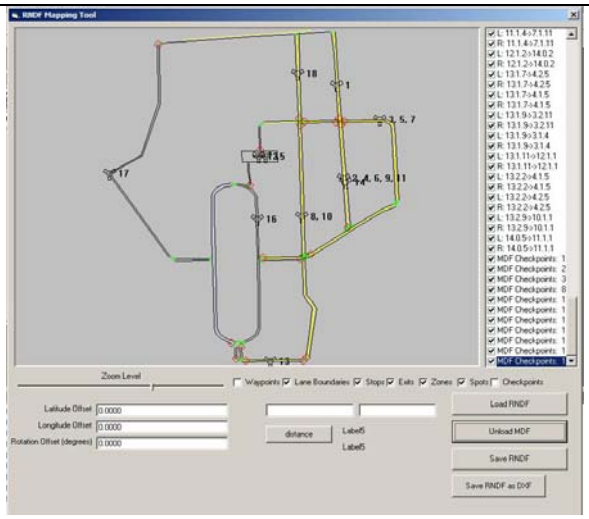
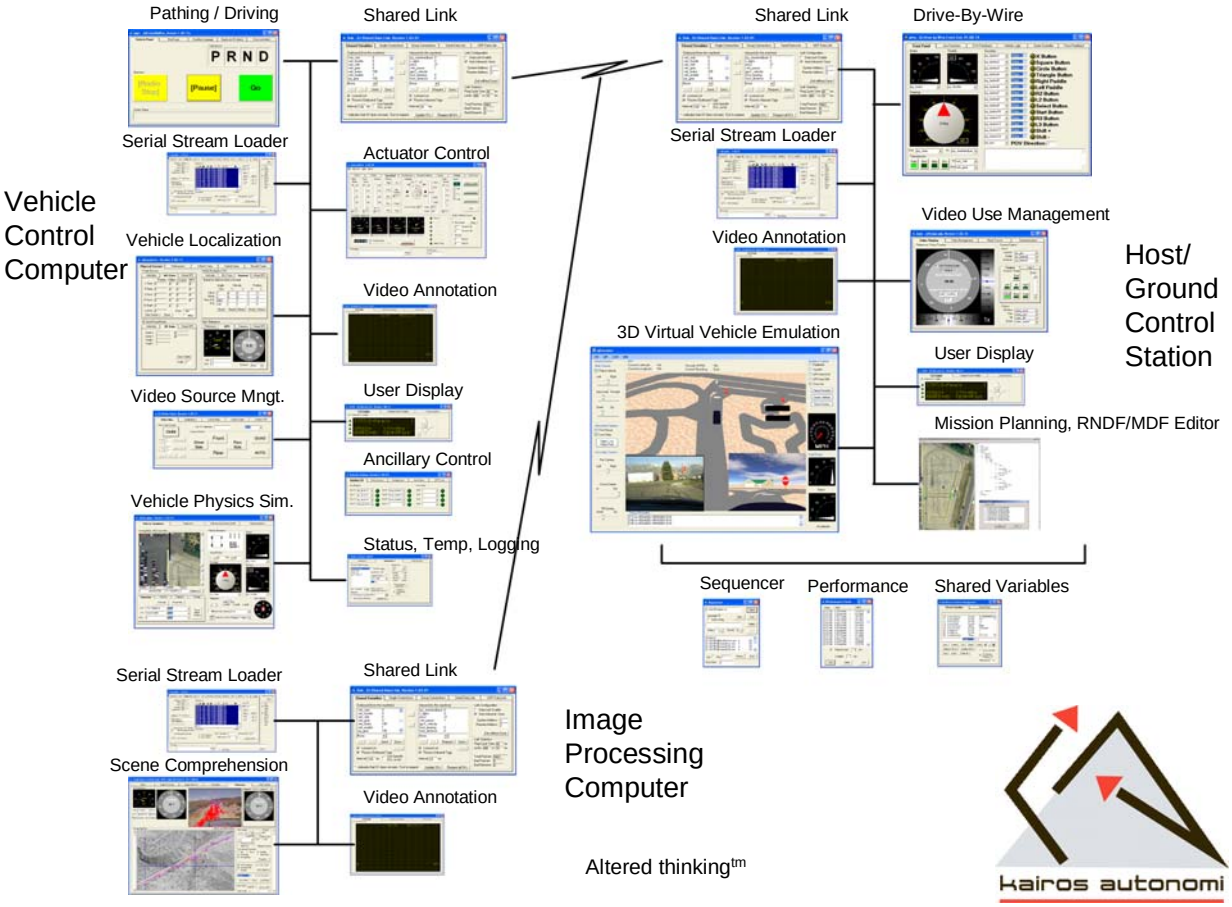


Figure 12, the **RNDF/MDF Graphical Editor** with GIS Synchronization used to create and manage test courses as well as the primary database used to store the RNDF and MDF on the vehicle. Images, left, of the DARPA supplied RNDF and MDF and, above, of Skyline High School driver training range.



All of the programs described in this technical document fit within the above frame work. This framework allows for the operation of our autonomous ground vehicle in a number of modes from simple drive-by-wire to full autonomy.

3. RESULTS AND PERFORMANCE

3.1 Risk Mitigation

In addition to continual program management, four measures are utilized to assure that risk is kept to a minimum during the entire development cycle.

Computing Performance: Performance measurement is continuously calculated and recorded through calculation of the whetstone algorithm. At a system level, whetstones per second are calculated every 10 seconds for about one second with remaining computing power. This performance value is affected by software, hardware health and temperature. Changes in this value are directly related to changes in hardware or software of the system.

"Thoughtless" Recording: Many parameters are recorded each time the system is run without any need for human intervention. When a problem is noticed, the already recorded data can be reviewed for first occurrence and cause.

Acceptance Test Plans: The continuous execution of the ATP not only assures proper operations for testing and usage of the vehicle, it assures that the vehicle's systems are functional over the long term of the project.

Multiple Prototypes/Deliverables: The creation and continuous migration of multiple prototypes and deliverable systems assures that there is not downtime associated with device failure and provides a sequence for the migration of newly completed technologies through the numbered functional systems. The sheer number of hours on these units helps to assure robustness.

In addition to the above unique mitigations, standard IT security and safekeeping practices are employed to assure preservation of the technology.

3.2 Test Facilities

DesignJug has a fully functional and equipped autonomous vehicle technology development laboratory, as well as warehouse space where static sensor development and testing occurs. In addition, DesignJug uses three different sites for testing:

- Skyline High School Driving Range, Salt Lake City, Utah — this is a driver's education course designed to test all aspects of vehicle operation by new, human drivers. The site is about three acres with a fence and K-rail on three sides. An observation tower is nicely located in the center on one side of the course. The course is small with the longest straight away about 350 feet, but very contained and performs well for navigation testing and limited traffic testing.
- West Valley City Driver License Office, West Valley City, Utah — a large driving range in a controlled area. The range has multiple intersecting roads with stop lights, signs and striping. It also has a railroad crossing and other urban road features, making it an ideal location for testing in an urban setting.
- Bonneville Seabase, Grantsville, Utah — a free form, 16 square mile area where DesignJug can drive anywhere. Using aerial photos, DesignJug laid down the 2.2 mile Grand Challenge NQE course on this test site and ran it daily. However, the site does not have a tunnel. It is fenced and well away from any human or vehicular traffic, with zero penalty for going off course. The team usually tests multiple vehicles at once with several test teams. Testing here is informal, allowing software folks the ability to try stuff without a time or formality hit. DesignJug performed significant Grand Challenge testing at this site and it hosted DARPA's site visit.

3.3 Test Documentation

Proper and adequate testing is the only way to assure that this technology achieves the planned results. An Acceptance Test Plan (ATP) has been created for each autonomous ground vehicle that will be used for testing. The ATP consists of a written procedure and a test record, and it is executed almost on a continuous basis as an overall regression test to assure proper vehicle operation prior to testing of new software or hardware. The ATP is continuously evolving as it is revised (added to, edited) upon the discovery of features or performances that do not meet

expectations. The ATP documents weak points or areas prone to failure in order to assure function prior to moving forward. The ATP is not a 100 percent test since its performance goal is less than 30 minutes, but it does test functions and features that are the end results of many properly operating sub-functions. In theory, the revised ATP just prior to delivery (race) covers 100 percent of the system (directly or indirectly). The ATP is performed on an informal and formal basis: informally used during development cycles to assure that the vehicle is in a known state before new features are introduced (informal ATPs are not recorded); and formally before loading for each field test, after unloading for a field test (in the field) and before formal demonstrations (each formal ATP is recorded on an ATP Test Record).

3.4 Testing Process

Team Juggernaut tests continuously, whether it's in the lab or in the field. The team uses testing metrics to evaluate whether an addition is improving the performance and function of the system. These metrics can be run virtually or in real-time, allowing team engineers the ability to test from the lab before testing in the field.

In order to assure that each field test occurs with the most chance of success, a fairly rigid process is followed. A test objectives document was created which includes the test site, test staff, noted changes to the vehicle system and objectives of the test. These objectives are used to determine the success or failure of the test and as a communications tool to the test team. The test objectives document also becomes the test record. The ATP is executed prior to stowing and loading the vehicle on the trailer. A stow/load procedure and checklist is executed to assure that the vehicle is properly prepared for travel. A test record is filled out for the ATP and the stow/load. After traveling to the test site, the vehicle is unloaded using a test/unload procedure. A full ATP is then executed to assure proper vehicle and system operation. A test record is filled out for the ATP and the test/unload. Prior to testing a safety briefing is performed by the test team leader or assignee. The safety briefing follows a safety briefing checklist.

3.5 Recent Test Results

This document was required by DARPA as a precursor to the Site Visit. In recent weeks and months the following has been tested in a tier approach leading up to the site visit and beyond.

1. Speed independent GPS path following
2. Stop and go traffic operation with leading vehicles, stop bars
3. Fourway stop precedence
4. Dynamic re-pathing while underway
5. Obstacle detection, classification and management
6. Road marking detection and management
7. Unplanned collision avoidance execution
8. Canned routine execution – U-turn, curve based turn, entry and exit of parking stall
9. Reverse operations
10. Omni-detection overview of traffic situations.

These tests have been perfomed with varing levels of success. When a test does not go as planned or the results are different than expected that task is divided into successful and not-so-

successful components. Those components are then tested separately and revised until success is achieved. With this approach, all testing ultimately is successful as that is what is required to move the program forward.

4. FUTURE DEVELOPMENT

As the program progresses the conditional logic nature of the systems is being migrated into adaptable scripting languages such as AVIL (Autonomous Vehicle Interpretive Language) and FIRE (Forth Inspired Rule Engine). These will allow the behavior of the vehicle to be modified on the fly. The system will then begin to use rule schedules to achieve the behavior required for the desired functionality based upon the perceived situation.

Since all software functionality is basically executable components, these will be enhanced from their existing performance levels on an individual basis. This modular approach to software development with multiple executables running under a common thread manager lets each module improve independently all bounded by the ATP.

5. CONCLUSION

With a proven hardware platform and continued software and sensor advancements, DesignJug is confident it can competitively compete in the 2007 Urban Challenge. The team's design approach and its commitment to daily testing allow for a high level of interoperability among the autonomous ground vehicle's various systems and components. These performance levels will continue to increase as the team accelerates its development and testing over the coming months.

DesignJug recognizes that significant obstacles remain before the team can achieve its goals, but, with well-defined benchmarks established and the team's proven ability to integrate additions to the vehicle, these obstacles can be overcome. DesignJug looks forward to these challenges in preparation for the Urban Challenge.

We never turn the vehicle off, it physically drives autonomously every day.

References

[1] The Defense Advanced Research Projects Agency (DARPA) Web site, http://www.darpa.mil/grandchallenge/overview.asp, retrieved May 23, 2007, on the World Wide Web.